Microservices com Python

@diegorubin

Onde posso ser encontrado

diegorubin

@diegorubin

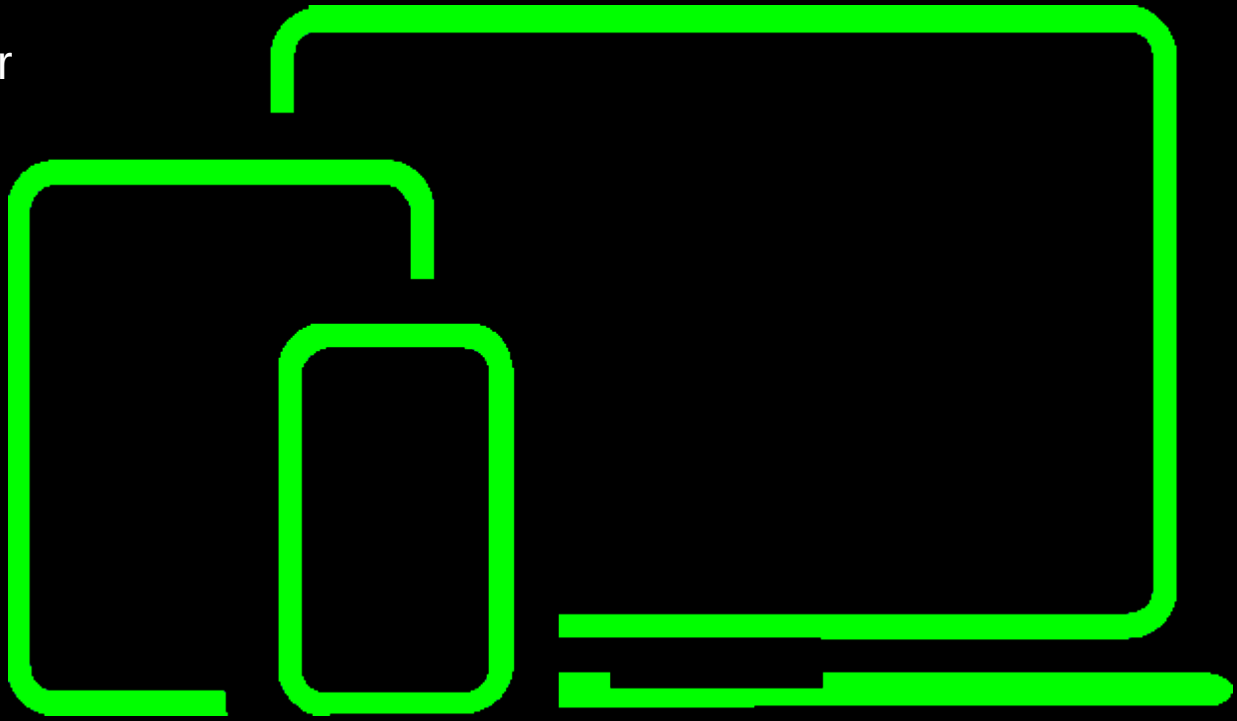
http://diegorubin.com

Como construímos software?

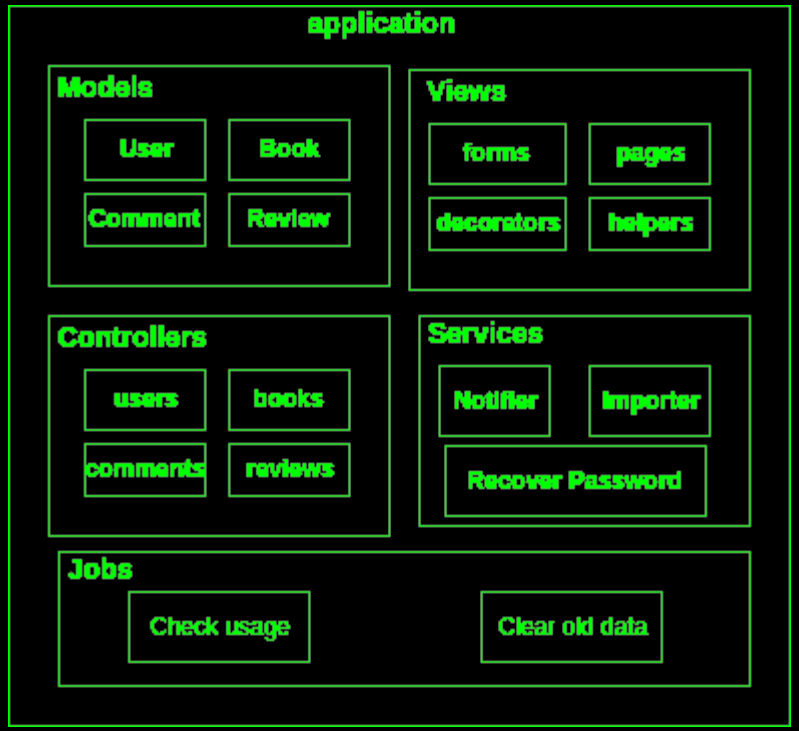
Como construímos sistemas?

Como construímos software

- Padrões de Projeto
- Models View Controller
- Aplicações para WEB
- Mobile First
- User eXperience



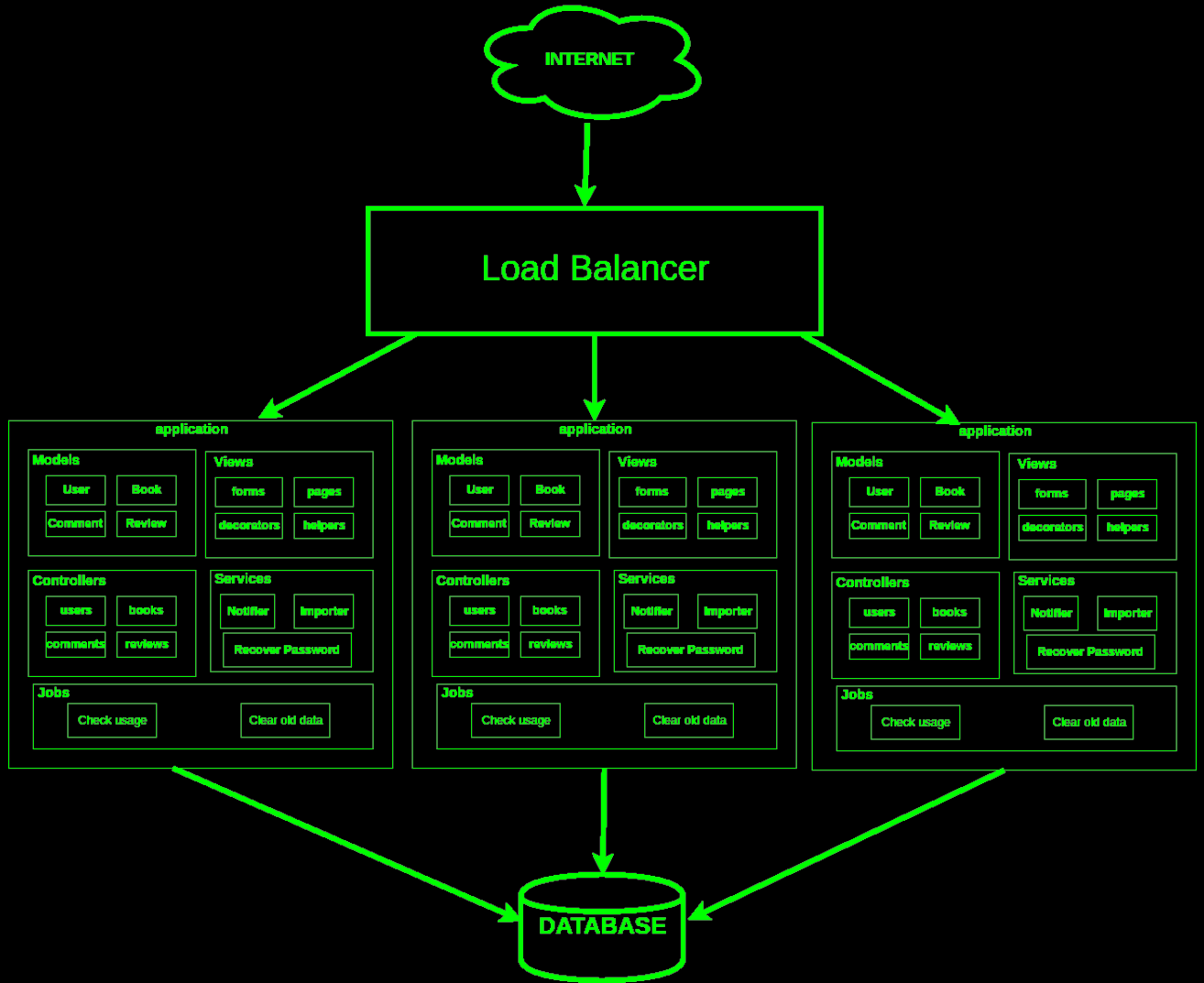
Design da Aplicação



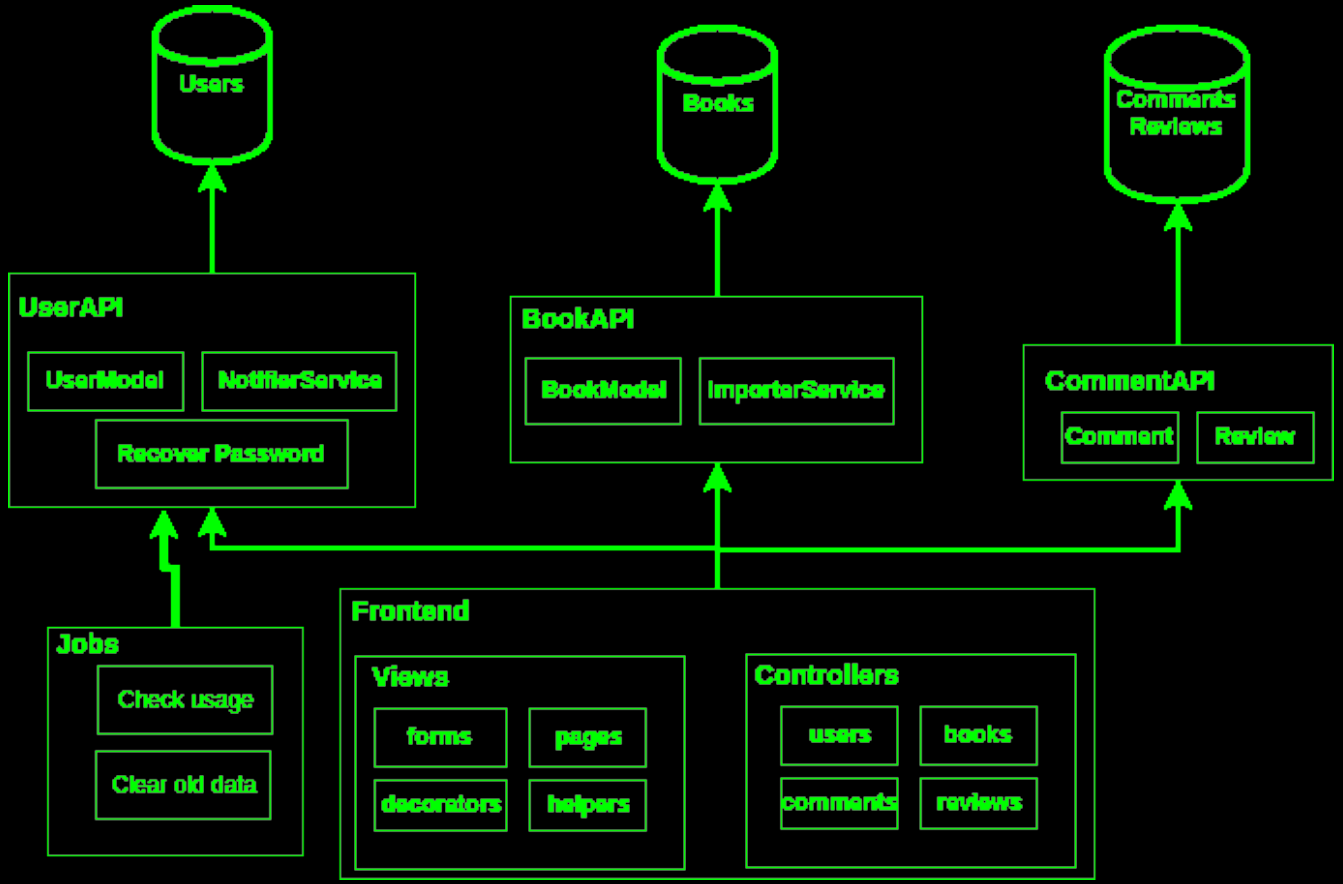
Esse modelo é chamado de monolítico.

Exemplos:

- Wordpress
- Magento
- Hybris


INTERNET
Load Balancer
application
Models
User
Book
Comment
Review
Views
forms
pages
decorators
helpers
Controllers
users
books
comments
reviews
Services
Notifier
Importer
Recover Password
Jobs
Check usage
Clear old data
application
Models
User
Book
Comment
Review
Views
forms
pages
decorators
helpers
Controllers
users
books
comments
reviews
Services
Notifier
Importer
Recover Password
Jobs
Check usage
Clear old data
application
Models
User
Book
Comment
Review
Views
forms
pages
decorators
helpers
Controllers
users
books
comments
reviews
Services
Notifier
Importer
Recover Password
Jobs
Check usage
Clear old data
DATABASE

E modelo microservices?



Vantagens desse modelo

- Desacoplamento
- Facilidade no deploy de novas features
- Uso de diversas tecnologias
- Facilidade em escalar a aplicação
- Falhar rapidamente, reagir rapidamente
- Facilmente testável
- Monitoração mais simples

Serviços REST

- Comunicação cliente-servidor sem estado
- Conjunto de operações bem definidas
- Identificadores únicos para os recursos
- Uso de hipermídia para representação

Serviços REST - trocado em miúdos

- Comunicação cliente-servidor sem estado
  - HTTP, HTTPS
- Conjunto de operações bem definidas
  - GET, POST, PUT, PATCH, DELETE, HEAD
- Conjunto de estados (https://httpstatusdogs.com/)
  - 200, 201, 404, 422, 500
- Identificadores únicos para os recursos
  - URI
- Uso de hipermídia para representação
  - JSON

JavaScript Object Notation - JSON

- Formato de dados
  - Nulo
  - Números
  - String
  - Boolean
  - Lista
  - Objetos

JSON - Exemplo

- Formato de dados
  - Nulo
  - Números
  - String
  - Boolean
  - Lista
  - Objetos

```
{
  “string”: “a string”,
  “objeto”: {
    “lista”: [true, 1.23, null]
  }
}
```

Recursos

uri: http://servidor/api/users

method: GET

request body: empty

response status: 200

response body:

```
[
  {“login”: “user1”, “email”: “user1@mail.com”},
  {“login”: “user2”, “email”: “user2@mail.com”}
]
```

Recursos

uri: http://servidor/api/users

method: POST

request body: {

“login”: “user3”,

“email”: “user3@mail.com”

}

response status: 201

response:

{

“uid”: “tralala”,

“login”: “user3”,

“email”: “user3@mail.com”

}

Recursos

uri: http://servidor/api/users/tralala

method: GET

request body: empty

response status: 200

response:

```
{
  “uid”: “tralala”,
  “login”: “user3”,
  “email”: “user3@mail.com”
}
```

Recursos

uri: http://servidor/api/users/tralala

method: PATCH

request body: {

“email”: “user3@mail.org”

}

response status: 200

response:

{

“uid”: “tralala”,

“login”: “user3”,

“email”: “user3@mail.org”

}

Recursos

uri: http://servidor/api/users/tralala

method: DELETE

request body: empty

response status: 204

response: empty

Para brincar

Serviços do Google -> https://developers.google.com/

Wunderlist -> https://developer.wunderlist.com/documentation

Marvel API -> https://developer.marvel.com/

ESPN -> http://www.espn.com/static/apis/devcenter/overview.html

SWAPI -> https://swapi.co/

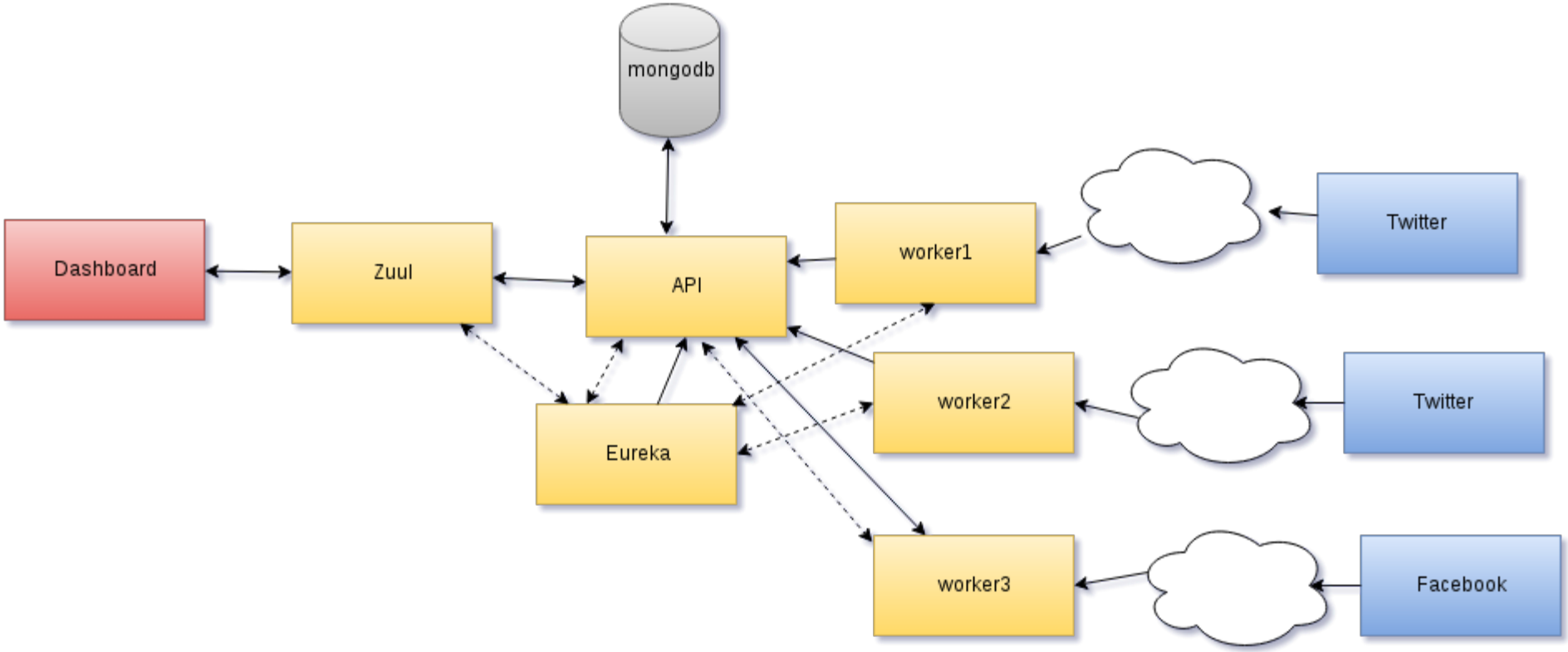
Procure por outras API’s -> http://99apis.com/home

Mensageria

- Fluxos assíncronos
- Controle de reprocessamento
- RabbitMQ, Apache Kafka
- Consumidores

Service Discovery

- Forma de notificar um serviço
- Facilitar configuração dos clientes
- Client-side load balancer
- Netflix Eureka, Consul


mongodb
Dashboard
Zuul
API
worker1
Twitter
worker2
Twitter
Eureka
worker3
Facebook

API Gateway

- Segurança
  - Autenticação
  - Throttle Control e Rate Limit
- Composição de API
- Load balancer e roteamento
- Netflix Zuul, Kong

Monitoração e Coleta de Logs

- Uso de healthchecks
  - Status de integrações
  - Serviços ativos
- Serviços de monitoria
  - Zabbix
  - Nagios
- Centralização e busca em logs
  - Graylog
  - Kibana
  - Splunk

Continous Delivery

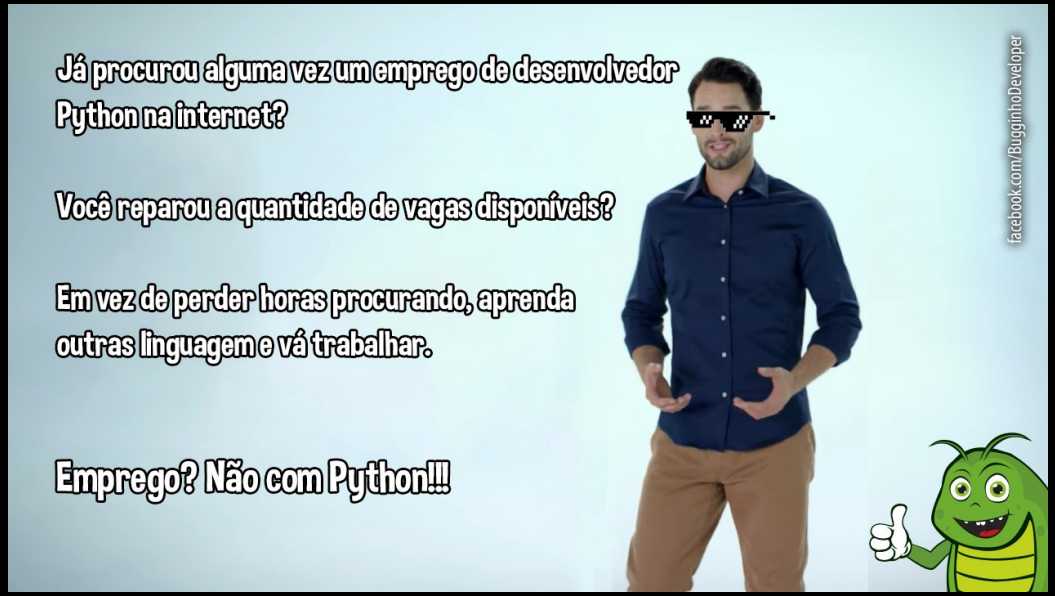
- Modelo antigo:
  - Especificação -> desenvolvimento -> Implantação -> Uso
- Scrum
  - Ciclo (Planning -> desenvolvimento -> demonstração -> implantação -> uso?)*
- Kanban
  - Ideia de fluxo contínuo
  - Não existe um ciclo
  - Subir ou não uma feature é uma decisão de negócio e não uma janela técnica
- Continous Delivery
  - Todo código commitado pode subir para produção
  - Todo código está na branch principal
  - Feature toggle
  - https://www.youtube.com/watch?v=dxk8b9rSKOo

Testes

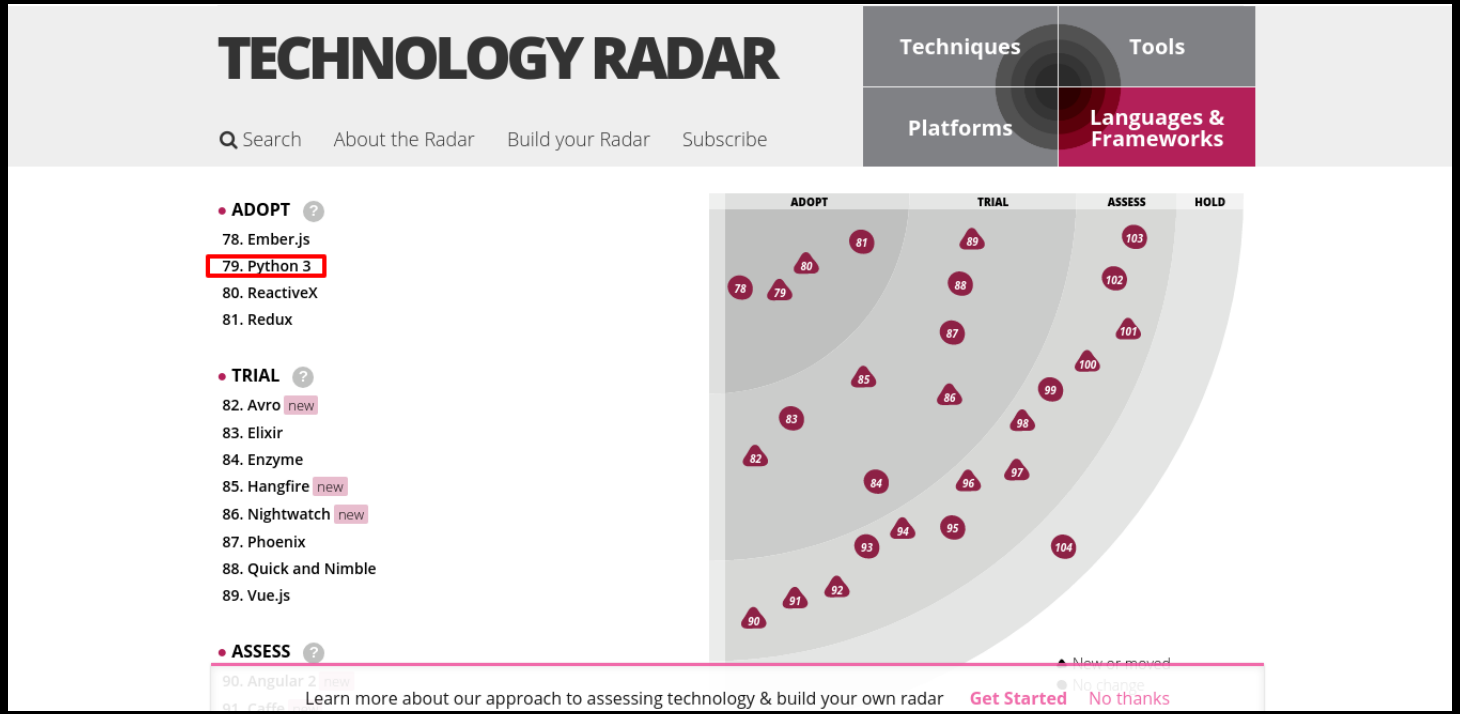
- Test Driven Development
- Mock
- Integration Tests
- Behaviour Driven Development
- Load Tests

Python?


16/11/2016 16h07 - Atualizado em 16/11/2016 16h07
Programadores Python visitam orfanato nessa quarta-feira. "É de cortar o coração ver aqueles rostos sem esperança", Disse Ana, de 6 anos.
python
facebook.com/BugginhoDeveloper


Já procurou alguma vez um emprego de desenvolvedor Python na internet?
Você reparou a quantidade de vagas disponíveis?
Em vez de perder horas procurando, aprenda outras linguagem e vá trabalhar.
Emprego? Não com Python!!!
facebook.com/BugginhoDeveloper

Sim, Python!



https://assets.thoughtworks.com/assets/technology-radar-vol-16-pt.pdf

Sim, Python!

Python é uma linguagem que continua aparecendo em lugares interessantes. Sua facilidade de uso como uma linguagem de programação geral, combinada com a sua forte base na computação matemática e científica tem historicamente levado a sua adoção pelas comunidades acadêmicas e de pesquisa. Mais recentemente, as tendências da indústria em torno da comoditização e das aplicações da AI, combinadas com a maturidade de Python 3, ajudaram a trazer novas comunidades para Python.

Sim, Python!

Abordagens arquitetônicas como microsserviços e contêineres facilitaram a execução de Python em ambientes de produção. Engenheiros podem agora implantar e integrar códigos especializados em Python criados por cientistas através de APIs agnósticas de linguagem e tecnologia. Esta fluidez é um grande passo em direção a um ecossistema consistente entre pesquisadores e engenheiros, em contraste com a existência de fato de traduzir linguagens especializadas como R para os ambientes de produção.

Python - Características Gerais

- Criada por Guido Van Rossum em 1991
- De propósito geral
- Multiplataforma
- Interpretada
- Tipagem dinâmica e forte
- Multi-paradigma
  - Orientação a Objetos
  - Funcional
  - Procedural e Imperativo
  - Reflexão (metaprogramação)
- PEP’s https://www.python.org/dev/peps/

Python - Tipagem dinâmica e forte

```
num = 5
print(num) # 5
print(num / “2”) # TypeError: unsupported operand type(s) for /: 'int' and 'str'
print(num / int(“2”))  # 2
print(num / float(“2”)) # 2.5
lista = [1, 3.14, ‘string’] # mutável
tupla = (1, 3.14, ‘string’) # imutável
dict = {‘a’: 1, 5: ‘string’}
dict[‘c’] = 2
```

Python - Sintaxe

```
if <condition>:
    pass
elif <condition>:
    pass
else:
    pass
```

Operadores lógicos

and, or, not, in

==, !=, <, <=, >, >=

```
for i in iterator:
    print(i)

while <condition>:
    if <condition>:
        break
```

```
def f1(arg1, arg2):
    return arg1 + arg2

def f2(arg1=1, arg2="2"):
    pass

f1(1, 2)
f2(arg2="3", arg1=2)
f2()
```

Python - Sintaxe

```
class Pai():
        def __init__(self, v):
                self.v = v
        def m(self, a):
                self.a = a
        @staticmethod
        def static():
                print('static')

p = Pai('v')
p.v # 'v'
p.m(1)
P.static()
```

```
class Filho(Pai):
        def m(self, a):
                print('filho')
                super(Filho, self).m(a)

f = Filho('v')
f.v # 'v'
f.m(1)
F.static()
```

Python - Sintaxe

```
try:
	pass
except Error as e:
	pass
except:
	print(‘qualquer outro erro’)
else:
	print(‘nao deu erro’)
finally:
	print(‘sempre passa por aqui’)

raise Error(‘teste’)
```

Python - Sintaxe

```
import caminho.do.modulo.funcao
import caminho.do.modulo.Classe
import caminho.do.modulo.CONSTANTE

caminho.do.modulo.funcao()
c = caminho.do.modulo.Classe()

from caminho.do.modulo import funcao
from caminho.do.modulo import Classe
from caminho.do.modulo import CONSTANTE

funcao()
c = Classe()
r = COSNTANTE * 2
```

Tornado

- Framework WEB
- Foco em performance
- Simples e com vários exemplos de uso
- Possui utilitários
  - cliente http
  - facilitadores para testes
  - integração com oauth

Tornado - Exemplo

```
import tornado.ioloop
import tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")

def make_app():
    return tornado.web.Application([
        (r"/", MainHandler),
    ])

if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()
```

Para executar
$ python3 exemplo.py

Documentação

- https://docs.python.org/3/
- http://www.tornadoweb.org/en/stable/